Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Contrastive Learning using SPECTRE for Generating Graph Augmentations

Semester Thesis

York v. Schlabrendorff

vyork@ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Karolis Martinkus, Zhao Meng
Prof. Dr. Roger Wattenhofer

March 2, 2023

Acknowledgements

I would like thank my supervisors Karolis Martinkus and Zhao Meng, whose unwavering support and guidance were instrumental in the successful completion of this semester project. Throughout the process, they were always reachable, taking the time to brainstorm dead-ends, innovative solutions and help me analyze subtle bugs, demonstrating not only technical expertise but also a true passion for their work. Without their assistance, this thesis would have never been accomplished. And especially due to their dedication, working on this project was an incredibly rewarding experience.

Abstract

In this thesis we explored to utility of SPECTRE for graph augmentations for a Contrastive Learning (CL) with subsequent downstream classification task. We investigated factors influencing the generated augmentations and analyzed their effect on the learning embeddings. Our classification results look promising being on par with supervised baselines, but analysis of the generated augmentations suggest that performance is not exhausted, yet and can be further optimized.

Contents

Chapter 1

Introduction

Graphs are an attractive datastructure for their versatility and ability to model complex relationships in a wide variety of domains, such as social networks, where nodes represent users and edges represent their connections, knowledge graphs, where nodes represent entities and edges represent relationships between them, molecular graphs, where nodes represent atoms and edges represent chemical bonds, and fraud detection in financial systems, where nodes represent transactions and edges represent the flow of money between them. As in any Machine Learning domain labeling data is expensive and at bottleneck for many projects. Self-Supervised Learning methods, and specifically Contrastive Learning as proposed in SimCLR [1], tries to alleviate this issue by learning the features of a dataset without the need of human-annotated labels. These pretrained embeddings are subsequently finetuned on downstream, achieving comparable performance with less labels.

For CL a model generates two versions of an original data sample by applying augmentations on it and then minimizing the distance between the embeddings of these two versions. These augmentations should add noise to image, without altering the features. For images augmentations are easily imaginable like cropping, resizing, scaling, rotations, distortions, but should not change the core properties by e.g. changing a cat to a dog. Fundamental to the success of CL are the augmentations, but for graphs less straightforward. Especially since for depending on the application e.g. the removal of a random node could either be insignificant or label altering.

This is were our research comes in: In this paper we research the utility of SPECTRE [2] to generate graph augmentations for Contrastive Learning. We propose a Contrastive Learning architecture inspired from Barlow Twins [3] in Chapter 2, evaluate the model on a downstream task and develop methods to analyze the performance of the augmentations in Chapter 3.

Chapter 2

Method

In this section we introduce our Contrastive Learning method, which consists of the three building blocks: the augmentation generation, the embedding, and the loss.

Contrastive Learning Setup. The architecture setup in this work draws inspiration from Barlow Twins (BT) [3] and is adapted to the graph neural network domain. As shown in Figure 2.1, the architecture consists of three phases: Augmentation Generation, Embedding, and Loss. First, based on an original graph as input, we generate two augmentations (Y^A and Y^B). Next, we embed both augmented graphs to obtain two corresponding embedding vectors (Z^A and Z^B). To encourage the embeddings to capture useful information about the input graph, while invariant to the augmentations, we calculate their cross-correlation and aim to make it resemble the identity matrix. In the following paragraphs, we provide a detailed discussion of each of these three steps.

(1) Augmentation. To generate augmentation we utilize the one-shot graph generator SPECTRE [2]. For our use-case a one-shot generator is favorable over autoregressive generators as it results in more novelty and is fully-parallelizable (faster training).

SPECTRE is able to capture the structure of graphs and generate new graphs belonging to the same distribution, by conditioning the generated graphs on spectral properties. This faithful reconstruction of graph properties is important for

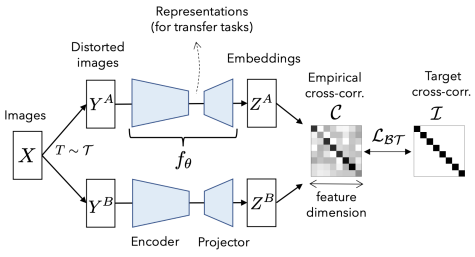

Figure 2.1: Excerpt from Barlow Twins [3] illustrating the training pipeline

CL models.
To condition a graph generation on the properties of an input graph with adjacency matrix A, SPECTRE computes the normalized Laplacian $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, where D is the diagonal degree matrix and I the identity. Then, an eigenvalue decomposition $L = U\Lambda U^T$ is performed on the graph Laplacian L, where $U = [u_1, ..., u_n]$ and $\Lambda = \text{diag}(\lambda_1, ...\lambda_n)$ correspond to the eigenvectors and eigenvalues resp.. These eigenvalues are sorted in ascending order, where the first, resp. lowest, contain the most information of the graph structure and are thus the most influential. Since we don't want to completely replicate the original graph, but want to generation alterations of it, we restrict the number of eigenvalues/eigenvectors passed to SPECTRE to the **k** lowest to support imperfect reconstruction within reason. Hence, we can tune the strength of conditioning and therefore the amount of alterations by parameter **k**: The higher the **k**, the more do the generations resemble the original graph, and vice versa. Note, that this does not directly steers the generation towards augmentations, which are sensible w.r.t. the dataset domain. During implementation, we noticed that PyTorch's batched eigenvalue decomposition pads the eigenvectors on the left side with zeros, when the batched graphs contain different number of nodes. The network struggled to handle that the most consequential inputs would jump around so that we had to align them to the left side with padding on the right so that λ_0 and u_0 are always on the zero-th index.

In the original SPECTRE paper the distribution of spectra in the dataset is also learned, so that the generated graphs are conditioned on in turn generated eigenvalues/eigenvectors, but we restricted our experiments to supplying SPECTRE with the real spectra.

To construct an augmentation $A', N', F' = g_A(\lambda_k, U_k, w_A)$ SPECTRE utilizes an l-layer Provably Powerful Graph Network (PPGN) [4], where A', N', F' are the generated adjacency (N × N × 2,) node features (N × F), and edge features (N × N × F) matrices. N is the number of nodes and F the number of node/edge features respectively. To enable diversity a latent variable z_A is sampled $w_A = \text{MLP}_{w_A}(z_A)$ and processed by an MLP. Note, that in the standard implementation there is no conditioning on node and edge features. Implications of that are discussed in Section 3.1.

(2) Embedding. For a graph embedding we used the Graph Isomorphism Network (GIN) [5] and adapted the implementation of OGB [6], as it supported node & edge feature embedding. Since this GIN implementation expects sparse graphs in a PyG mini-batch format, we had to pre-embed the node and edge features. For that we discretized the prediction logits into one-hot vectors using the Gumbel-Softmax [7] trick (since an argmax would destroy the gradients) and multiplied them with the weights of the respective node (atom) and edge (bond) encoders. We used 5 GIN layers with an embedding dimension of 300, mean pooling and an embedding dropout of 0.5, which were the default values used in

OGB [6].

(3) Loss. The loss consists of two important parts: (1) a Reconstruction Loss trying to minimize the distance to the original graph and (2) a Barlow Twins Loss pioneered by [3], which steers the network to learn effective representations.

The Reconstruction Loss is fundamental to the networks ability to resemble the original graphs. Without it the network would collapse to always predicting full graphs as the Barlow Twins loss does not encourage not predicting an edge. The reconstruction loss applies the predicted, dense logits to three Cross-Entropy losses for each of the adjacency, node and edge features matrices respectively together with the original graph. For this the original graph has to be converted to a dense representation.

The BT loss on the other hand does not contrast each augmentation with the original graph, but the two augmentations with each other. Earlier self-supervised learning methods struggled with mode collapse to constant trivial solutions [3]. BT avoids this by measuring the cross-correlation between the augmentations and making it as close to the identity matrix as possible. An identity cross-correlation would mean that each feature is independent of each other, resulting in maximal expressiveness and utilization of the networks capacity. The diagonal elements of the cross-correlation matrix (supposed to be 1) make the embedding invariant to augmentations applied, while the off-diagonal elements (supposed to be 0) decorrelate the embedding.

We deviated from the BT reference implementation in two points. First, we noticed that the magnitude of the loss formulated in the original paper depended on the size of the inputted vectors. This caused difficulties when combining it with the reconstructive loss terms, so that we normalized the diagonal and off-diagonal sums by their number of elements (essentially resulting in a MSE). Consequently, we deviated from their proposed loss trade-off of 0.005 to an equal summation of on-diagonal and off-diagonal loss. Secondly, we disabled the tracking of running stats in the Batch Norm layers.

The total loss is summed as

$$\mathcal{L} = \text{R-TO} \cdot 2(\mathcal{L}_{Adj} + \mathcal{L}_{Node} + \mathcal{L}_{Edge}) + (1 - \text{R-TO}) \cdot (\mathcal{L}_{\text{on-diag}} + \text{B-TO} \cdot \mathcal{L}_{off\text{-}diag})$$

We did not trade-off the three reconstructive losses with each other, as this would have resulted in too many parameters to tune even though it would have been sensible to do. R-TO and B-TO stands for Reconstructive Trade-Off and Barlow Trade-Off respectively. Note that the multiplication factor of 2 is a simplification of the fact that the reconstructive loss is calculated twice, once for each augmentation.

Our best performing model was trained with $\mathbf{k} = 8$, a trade-off between the on-diagonal and off-diagnoal terms of the Barlow Twins Loss B-TO = 2, a reconstructive trade-off of R-OT = 0.1, an embedding dimension of 300, symmetric

Laplacian normalization, a batch size of 40 and for 30 epochs. Parameters that we did not experimented with were set to 4 PPGN layers, a latent dimension of 128 for noise z, 5 GIN embedding layers, and mean graph pooling.

2.1 Variations

We also experimented with a number of variations to this architecture, which did not immediately result in conclusive improvements.

Adversarial. We experimented with an adversarial setup similar to a Generative Adversarial Network. Conceptually the idea is persuasive: A risk when generating the augmentations is that they are too close to the original graph and would not capture applicable range of plausible augmentations. In our adversarial setup the optimizer during generation minimizes the reconstructive losses (to keep the augmentations plausible), while maximizing the BT loss to train the augmentations, which fool the embedding. During the embedding phase (analog to the discriminator phase in a classical GAN) the optimizer minimizes the BT loss. This should enable the network to be more robust to out of distribution samples.

Spectral Reconstructive Loss. Another approach to prevent augmentations being too close was to replace the Reconstructive Cross Entropy Loss on the adjacency matrix with a Spectral Reconstructive Loss. This followed from the observation that the the Cross Entropy losses penalized completely implausible edge prediction and plausible edge prediction equally. By computing the loss over the spectral properties we hope that the loss would become "less hard" and "fuzzier", penalizing spectral altering augmentations more. We formulated the Spectral Reconstructive Loss as

$$\text{MSE}(\text{diag}(U^T L_{Aug} U), \Lambda) \tag{2.1}$$

where L_{Aug} is the Laplacian of the generated augmentation and U and Λ are the real eigenvectors, resp. eigenvalues. Equation 2.1 is a reformulation of the eigenvalue decomposition

$$L = U\Lambda U^T$$
$$U^T L U = \Lambda$$

Unfortunately our implementation was not able to overfit to a single batch (each epoch the graphs would become sparser until eventually empty) so that we had to abandon this approach due to time constraints.

Conditioning on Node and Edge Features. As described above the graph generator is only conditioned on the graph spectrum and nothing else. Since the graph adjacency influences node and edge features only to a limited

degree, the graph generator struggles to learn these. A required remedy would be to additionally condition generated graphs on node and edge features. As a preliminary proof-of-concept experiment, we concatenate the real edge features with the initial Laplacian build before the first layer of the PPGN, to judge the effect on loss and information content.

Chapter 3

Experimental Evaluation

This chapter is divided into two sections: The first discusses the prediction performance, where the embeddings are finetuned to a downstream task, while the second seeks to explain observed behaviours by analysing the embeddings and augmentations.

3.1 Finetuning Results

Dataset. For this project we concentrated us on the `ogb-molhiv` dataset as developed by Hu et al. [6]. The task is binary classification of molecules, whether they inhibit the HIV virus replication or not. The dataset consists of 41,127 graphs with on average 25.5 nodes per graph. We ignored outlier graphs with more than 100 nodes and less than 18, hence resulting in 32,573 graphs split into train (64%), test (20%), and validation (16%) sets. The results discussed below should therefore be interpreted with caution as the evaluation was conducted on this subset of `ogb-molhiv`. As a result, the results may not be directly comparable to those obtained from the full dataset.

Setup. As a finetuning network we utilized an MLP with one hidden layer with half the embedding dimension and a single output neuron for binary classification. Since the dataset is skewed with roughly 23 negative samples for each positive samples, we increased the weight of positive weights in the Binary Cross Entropy loss accordingly.

Baselines. To put our results in context, we compare our results to three baselines. The first follows the suggestion of [8] to finetune a randomly initialized GNN embedding to establish a lower bound. Secondly, as our hypothesis is that SPECTRE augmentations suit the CL task better than trivial augmentations, we compare to simple augmentations, which rewire randomly 20% of the edges. The third baseline is the GIN with additional features, but without a virtual node as published in OGB [6] on their test set.

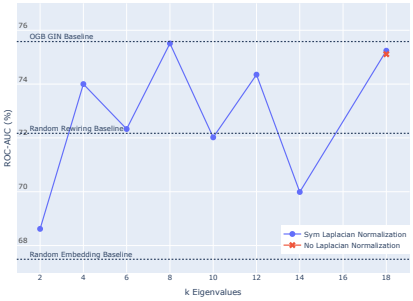

Figure 3.1: ROC-AUC vs. number of Eigenvalues for conditioning

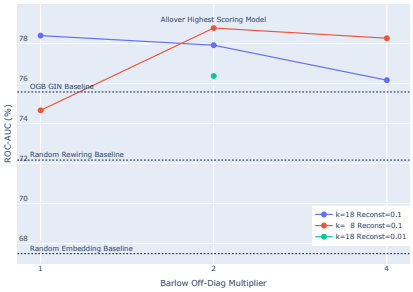

Figure 3.2: ROC-AUC vs. multiple settings of Barlow and Reconstructive Loss Trade-Off parameters

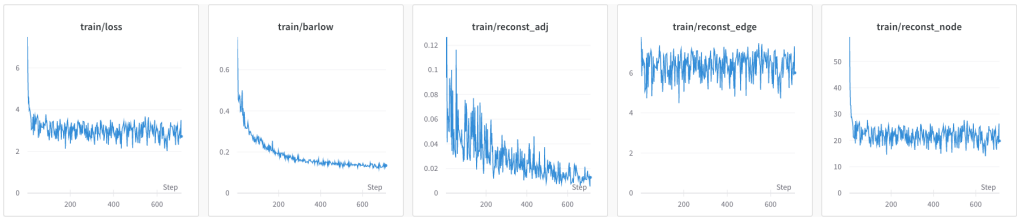

Figure 3.3: (1) Typical example for Magnitudes of the different loss terms (2) Conditioning on Edge Features has no visible impact on the reconstructive edge loss.

k Eigenvalues. Our initial hypothesis was that by tuning the reconstructional quality with **k** of SPECTRE and henceforth the amount of alterations, we would be able to influence the embedding quality and prediction performance of the network. Unfortunately, we cannot observe a clear relation ship between **k** and the prediction performance. All runs plotted in Figure 3.1 were performed with a BT trade-off of barlow $=$ on-diag $+ 0.5 \cdot$ off-diag and a reconstructive trade-off of $0.3 \cdot$ reconst terms $+ (1 - 0.3) \cdot$ barlow. Note, that the off-diagonal BT loss is not halved here intentionally, but due to an implementation bug. This should however not affect the comparison of different **k** values.

Trade-Off Parameters. It is difficult to find the right balance between in total 8 loss terms (2×3 reconstructive terms $+2$ BT terms). The example loss behaviour depicted in Figure 3.3 demonstrates the scale of each parameter, and is representative for all runs. A sweep over different parameters is shown in Figure 3.2. We found the all-over best scoring configuration for $\mathbf{k} = 8$, barlow $=$ on-diag $+ 2 \cdot$ off-diag, $0.1 \cdot$ reconst terms $+ (1 - 0.1) \cdot$ barlow. As we observe significant performance improvements after tuning the B-TO and R-OT parameters, it would be worthwhile to explore the impact of a finely tuned trade-off between the three reconstructive losses.

Embedding Dimension. The authors of the Barlow Twins paper [3] experienced increased performance on their task by using very high-dimensional embeddings. Hence, we also experimented with embedding dimensions 150, 300 (OGB default) and 600, scoring respectively $74.57\%, 75.51\%, 73.04\%$. Since the initial choice and OGB default of 300 performed best, we did not further explored this parameter.

Conditioning on Edge Features. As mentioned in Chapter 2, the graph generator is only conditioned on the graph spectrum and nothing else. This is most likely the reason why the networks fails to improve the node and edge reconstructive losses in Figure 3.3. They hovered around a constant value, meaning that the network most likely collapsed to constantly predict the mean of the distribution. To test, whether the network would be able to improve on these losses with additional information, we conditioned the run in shown in Figure 3.3 also on real edge features. In one initial proof-of-concept experiment we could observe a jump of prediction performance by 2.32 percentage points, but no improvement in terms of training loss. Though, including the *real* edge features runs contrary to the idea of generating augmentations.

3.2 Embedding & Augmentation Analysis

Cosine Similarity. The most important part of CL are the augmentations. To better understand and quantify the augmentation's performance we generated for every original graph 100 augmentations, embedded them, and calculated the Cosine Similarity for these 100 embeddings of the augmented vectors with the embedding of the original graph.

A Cosine Similarity too high would mean that the augmentations are too close to the original graph. Orthogonal embeddings on the other hand would suggest that the generated augmentations are out of distribution and not similar to the original. Another interesting metric is the support of the resulting probability distribution as it signifies the range of augmentations from similar to dissimilar. We hypothesize that the ideal Cosine Similarity distribution should hover around 0.5 to 0.8.

In Figure 3.4 we observed that our SPECTRE augmentations fall short of our goal to create realistic, in-distribution augmentations, since they are mostly orthogonal to slightly anti-correlated to the original graph's embedding. Nevertheless, the prediction performance seems to show that the embeddings capture enough features to slightly exceed the OGB Baseline (but nowhere near the top of the leaderboard, which scores 84.20%). This indicates that there is still a lot of potential performance left in this CL approach.

Intriguingly, while no effect of conditioning on edge features could be observed on the edge reconstruction loss during training, its Cosine Similarity distribution

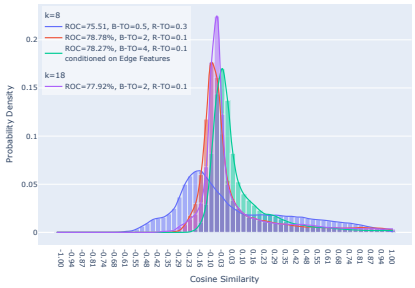

Figure 3.4: Distribution of Cosine Similarities between Augmentations and their Original w.r.t. Graph Embeddings

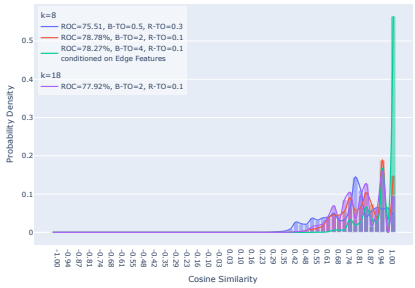

Figure 3.5: Distribution of Cosine Similarities between Augmentations and their Original w.r.t. Node Feature Embeddings

w.r.t. **node** feature vectors are clearly influenced, as depicted in Figure 3.5. The excessive Cosine Similarity scores highlight the necessity of abstracting the real features for feature conditioning. When real features are overly influential, they can constrain the augmentation process, leading to a shortage of variation in the augmented features.

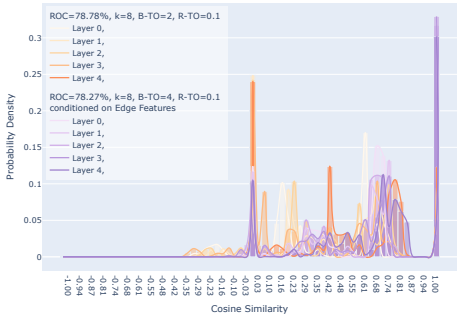

Figure 3.6: Distribution of Cosine Similarities between Augmentations and their Original w.r.t. Edge Embedding of each GIN Layer

Visual analysis of edge embeddings is more challenging compared to node embeddings as edge embeddings are calculated for each GIN layer. In contrast, node embeddings are only computed once at the beginning. Therefore, when plotting edge embeddings, one must consider the impact of multiple GIN layers on each edge embedding, which convolute the plots. An example is shown in Figure 3.6, depicting the difference between the best performing model and the model conditioned on edge features. A similar tendency as with the node features towards higher Cosine Similarity values can be observed here as well.

Influence of k Eigenvalues on generated Adj.

In Figures 3.7 to 3.14 the generated augmentations on the validation set at epoch 30 are shown (please zoom in). A couple observations can be made:

1. Figure 3.7: If the Reconstructive loss Trade-Off term (R-TO) is too high, no augmentations take place.

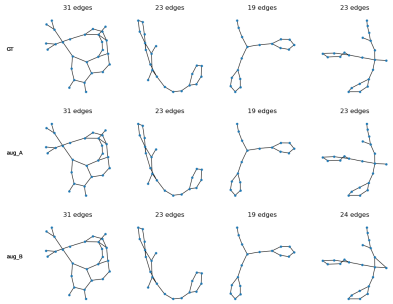

Figure 3.7: ROC=75.51%, B-TO=0.5, R-TO=0.3, k=8

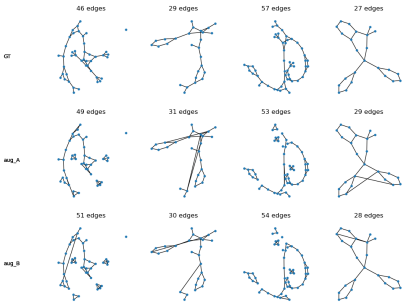

Figure 3.8: ROC=77.92%, B-TO=2, R-TO=0.1, k=18

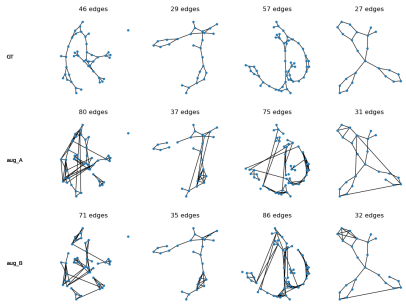

Figure 3.9: ROC=78.78%, B-TO=2, R-TO=0.1, k=8

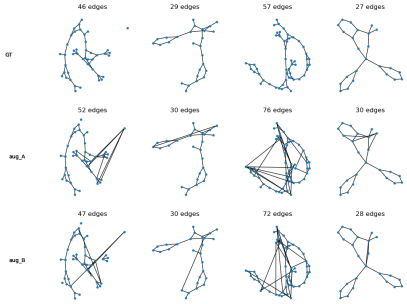

Figure 3.10: ROC=78.27%, B-TO=4, R-TO=0.1, conditioned on Edge Features, k=8

2. A higher reconstructive loss helps networks with low **k** to capture the high-level structure.

3. Augmentations with low **k** often contain filled cycles (i.e. all nodes in a cycle are connected).

4. Generally, augmentations always increase the total number of edges.

5. Figures 3.13, 3.14: Normalizing the Laplacian by the datasets maximal Eigenvalue, impedes faithful reconstruction heavily.

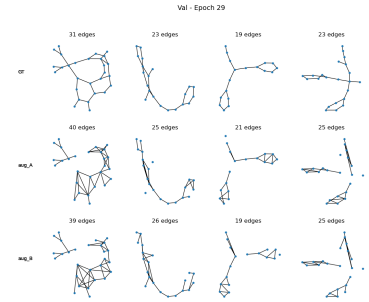

Figure 3.11: k=2, ROC=68.62%, B-TO=0.5, R-TO=0.3

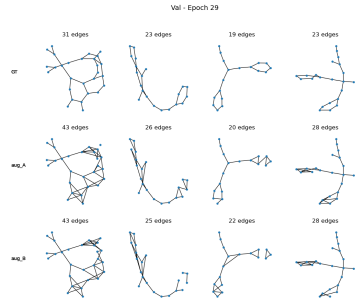

Figure 3.12: k=4, ROC=74.00%, B-TO=0.5, R-TO=0.3

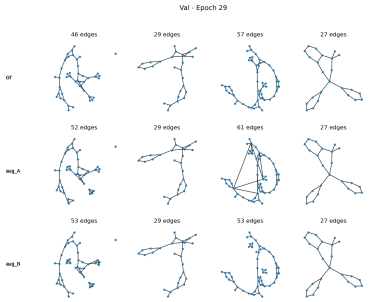

Figure 3.13: Symmetric Laplacian Normalization, ROC=75.24%, B-TO=0.5, R-TO=0.3, k=18

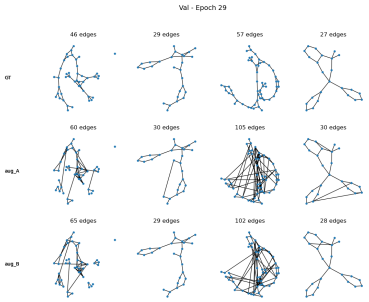

Figure 3.14: Laplacian Normalization by max Eigenvalue, ROC=75.11%, B-TO=0.5, R-TO=0.3, k=18

Chapter 4

Conclusion & Outlook

In conclusion, our results demonstrate that SPECTRE is a promising approach for generating augmentations and achieving strong performance on downstream tasks. Nevertheless, optimizing and fine-tuning it to specific domains is a non-trivial task that requires further research to fully understand its upper limits, as we suspect that there is performance left ot be gained.

Further research is especially needed for node and edge feature conditioning to carry over the SPECTRE's success generating adjacency augmentations to the generation of node and edge features.

Though a case can also be made to first perfect pure adjacency generation on a dataset, which does not contain node and edge features. Lower **k** values lead to undesirable artifacts as e.g. filled cycles. Conditioning on learned eigenvalues and eigenvectors, as proposed in the SPECTRE paper [2], could be a potential solution to retaining novelty with higher **k** values, while also minimizing artifacts. This in turn, could allow to evaluate whether our technique of comparing Cosine Similarity of embeddings and our hypothesis, which values would be ideal, are actually desirable.

Furthermore, it is not evident whether the Cross Entropy Reconstructive Loss we used is the ideal one. Since one of the desirable properties of augmentation generation is that realistic augmentations are generated, it could make sense to penalize clearly illegal graphs, e.g. in the molecular domain to physically impossible bonds. For example, carbon, which has four valence electrons, can form up to four covalent bonds with other atoms, such as in methane (CH4), but not more. Carbon nodes could therefore be penalized should their degree be greater than 4.

Bibliography

[1] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton, "Big self-supervised models are strong semi-supervised learners," *arXiv preprint arXiv:2006.10029*, 2020.

[2] K. Martinkus, A. Loukas, N. Perraudin, and R. Wattenhofer, "Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators," 2022. [Online]. Available: https://arxiv.org/abs/2204.01613

[3] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," 2021. [Online]. Available: https://arxiv.org/abs/2103.03230

[4] H. Maron, H. Ben-Hamu, H. Serviansky, and Y. Lipman, "Provably powerful graph networks," *CoRR*, vol. abs/1905.11136, 2019. [Online]. Available: http://arxiv.org/abs/1905.11136

[5] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2018. [Online]. Available: https://arxiv.org/abs/1810.00826

[6] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," *CoRR*, vol. abs/2005.00687, 2020. [Online]. Available: https://arxiv.org/abs/2005.00687

[7] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," 2016. [Online]. Available: https://arxiv.org/abs/1611.01144

[8] P. Trivedi, E. S. Lubana, Y. Yan, Y. Yang, and D. Koutra, "Augmentations in graph contrastive learning: Current methodological flaws & towards better practices," 2021. [Online]. Available: https://arxiv.org/abs/2111.03220